

Lenguaje de ruta XML (XPath) Versión 1.0

Recomendación del W3C del 16 de noviembre de 1999 (estado actualizado en octubre de 2016)

Esta versión:
http://www.w3.org/TR/1999/REC-xpath-19991116
(disponible en XML o HTML)
Ultima versión:
http://www.w3.org/TR/xpath
Versión anterior:
http://www.w3.org/TR/1999/PR-xpath-19991008
http://www.w3.org/1999/08/WD-xpath-19990813
http://www.w3.org/1999/ 07/WD-xpath-19990709
http://www.w3.org/TR/1999/WD-xslt-19990421
Editores:
James Clark <jjc@jclark.com>
Steve DeRose (Inso Corp. y Universidad de Brown) <Steven_DeRose@Brown.edu>

Abstracto

XPath es un lenguaje para abordar partes de un documento XML, diseñado para ser utilizado tanto por XSLT como por XPointer.

Estado de este documento

> *Actualización de estado (octubre de 2016):* aunque XPath 1.0 sigue utilizándose ampliamente y se hace referencia normativa en otras especificaciones del W3C, se informa a los lectores que existen versiones posteriores y que no se planea ningún mantenimiento adicional (incluida la corrección de errores informados) para este documento. Se recomienda a los lectores interesados en la versión más reciente de la especificación XPath que consulten https://www.w3.org/TR/xpath-3/ .

Este documento ha sido revisado por miembros del W3C y otras partes interesadas y ha sido respaldado por el Director como Recomendación del W3C . Es un documento estable y puede usarse como material de referencia o citarse como referencia normativa de otros documentos. El papel del W3C al elaborar la Recomendación es llamar la

atención sobre la especificación y promover su implementación generalizada. Esto mejora la funcionalidad y la interoperabilidad de la Web.

La lista de errores conocidos en esta especificación está disponible en http://www.w3.org/1999/11/REC-xpath-19991116-errata .

La versión en inglés de esta especificación es la única versión normativa. Sin embargo, para traducciones de este documento, consulte http://www.w3.org/Style/XSL/translations.html .

Puede encontrar una lista de las recomendaciones actuales del W3C y otros documentos técnicos en http://www.w3.org/TR .

Esta especificación es un trabajo conjunto del Grupo de Trabajo XSL y el Grupo de Trabajo de Enlace XML y por lo tanto es parte de la actividad Estilo del W3C y de la actividad XML del W3C .

Tabla de contenido

Apéndices

1. Introducción

XPath es el resultado de un esfuerzo por proporcionar una sintaxis y una semántica comunes para la funcionalidad compartida entre XSL Transformations [XSLT] y XPointer [XPointer] . El objetivo principal de XPath es abordar partes de un documento XML [XML] . Para respaldar este propósito principal, también proporciona funciones básicas para la manipulación de cadenas, números y valores booleanos. XPath utiliza una sintaxis compacta, no XML, para facilitar el uso de XPath dentro de URI y valores de atributos XML. XPath opera en la estructura lógica y abstracta de un documento XML, en lugar de en su sintaxis superficial. XPath recibe su nombre del uso de una notación de ruta como en las URL para navegar a través de la estructura jerárquica de un documento XML.

Además de su uso para direccionamiento, XPath también está diseñado para que tenga un subconjunto natural que pueda usarse para hacer coincidencias (probar si un nodo coincide o no con un patrón); este uso de XPath se describe en XSLT .

XPath modela un documento XML como un árbol de nodos. Hay diferentes tipos de nodos, incluidos nodos de elementos, nodos de atributos y nodos de texto. XPath define una forma de calcular un valor de cadena para cada tipo de nodo. Algunos tipos de nodos también tienen nombres. XPath es totalmente compatible con espacios de nombres XML [nombres XML] . Por lo tanto, el nombre de un nodo se modela como un par que consta de una parte local y un URI de espacio de nombres posiblemente nulo; esto se llama nombre expandido . El modelo de datos se describe en detalle en [**5 Modelo de datos**] .

La construcción sintáctica principal en XPath es la expresión. Una expresión coincide con la producción Expr . Se evalúa una expresión para producir un objeto, que tiene uno de los cuatro tipos básicos siguientes:

- conjunto de nodos (una colección desordenada de nodos sin duplicados)
- booleano (verdadero o falso)
- número (un número de punto flotante)
- cadena (una secuencia de caracteres UCS)

La evaluación de expresiones ocurre con respecto a un contexto. XSLT y XPointer especifican cómo se determina el contexto para las expresiones XPath utilizadas en XSLT y XPointer respectivamente. El contexto consta de:

- un nodo (el **nodo de contexto**)
- un par de números enteros positivos distintos de cero (la **posición del contexto** yel **tamaño del contexto**)
- un conjunto de enlaces variables
- una biblioteca de funciones
- el conjunto de declaraciones de espacios de nombres dentro del alcance de la expresión

La posición del contexto es siempre menor o igual que el tamaño del contexto.

Los enlaces de variables consisten en una asignación de nombres de variables a valores de variables. El valor de una variable es un objeto, que puede ser de cualquiera de los tipos posibles para el valor de una expresión, y también puede ser de tipos adicionales no especificados aquí.

La biblioteca de funciones consta de una asignación de nombres de funciones a funciones. Cada función toma cero o más argumentos y devuelve un único resultado. Este documento define una biblioteca de funciones principales que todas las implementaciones de XPath deben admitir (consulte [**4 Biblioteca de funciones principales**]). Para una función en la biblioteca de funciones principales, los argumentos y el resultado son de cuatro tipos básicos. Tanto XSLT como XPointer amplían XPath definiendo funciones adicionales; algunas de estas funciones operan en los cuatro tipos básicos; otros operan con tipos de datos adicionales definidos por XSLT y XPointer.

Las declaraciones de espacios de nombres consisten en una asignación de prefijos a URI de espacios de nombres.

Los enlaces de variables, la biblioteca de funciones y las declaraciones de espacios de nombres utilizados para evaluar una subexpresión son siempre los mismos que los utilizados para evaluar la expresión contenedora. El nodo de contexto, la posición del contexto y el tamaño del contexto utilizados para evaluar una subexpresión a veces son diferentes de los utilizados para evaluar la expresión que la contiene. Varios tipos de expresiones cambian el nodo de contexto; sólo los predicados cambian la posición y el tamaño del contexto (consulte [**2.4 Predicados**]). Cuando se describe la evaluación de un tipo de expresión, siempre se indicará explícitamente si el nodo de contexto, la posición del contexto y el tamaño del contexto cambian para la evaluación de subexpresiones; Si no se dice nada sobre el nodo de contexto, la posición del contexto y el tamaño del contexto, permanecen sin cambios para la evaluación de subexpresiones de ese tipo de expresión.

Las expresiones XPath suelen aparecer en atributos XML. La gramática especificada en esta sección se aplica al valor del atributo después de la normalización XML 1.0. Así, por ejemplo, si la gramática utiliza el carácter <, este no debe aparecer en el código fuente XML como <sino que debe citarse de acuerdo con las reglas XML 1.0, ingresándolo, por ejemplo, como <. Dentro de las expresiones, las cadenas literales están delimitadas por comillas simples o dobles, que también se utilizan para delimitar atributos XML. Para evitar que el procesador XML interprete que una comilla en una expresión termina el valor del atributo, se puede ingresar la comilla como referencia de carácter ("o '). Como alternativa, la expresión puede utilizar comillas simples si el atributo XML está delimitado por comillas dobles o viceversa.

Un tipo importante de expresión es una ruta de ubicación. Una ruta de ubicación selecciona un conjunto de nodos relativos al nodo de contexto. El resultado de evaluar una expresión que es una ruta de ubicación es el conjunto de nodos que contiene los nodos seleccionados por la ruta de ubicación. Las rutas de ubicación pueden contener de forma recursiva expresiones que se utilizan para filtrar conjuntos de nodos. Una ruta de ubicación coincide con la LocationPath de producción .

En la siguiente gramática, los no terminales QName y NCName se definen en [XML Names] y S se define en [XML] . La gramática utiliza la misma notación EBNF que [XML] (excepto que los símbolos gramaticales siempre tienen letras mayúsculas iniciales).

Las expresiones se analizan dividiendo primero la cadena de caracteres que se va a analizar en tokens y luego analizando la secuencia de tokens resultante. Los espacios en blanco se pueden utilizar libremente entre tokens. El proceso de tokenización se describe en [**3.7 Estructura léxica**] .

2 rutas de ubicación

Aunque las rutas de ubicación no son la construcción gramatical más general del lenguaje (un LocationPath es un caso especial de un Expr), son la construcción más importante y, por lo tanto, se describirán primero.

Cada ruta de ubicación se puede expresar utilizando una sintaxis sencilla pero bastante detallada. También existen una serie de abreviaturas sintácticas que permiten expresar los casos comunes de forma concisa. Esta sección explicará la semántica de las rutas de ubicación utilizando la sintaxis no abreviada. Luego se explicará la sintaxis abreviada mostrando cómo se expande a la sintaxis no abreviada (consulte [**2.5 Sintaxis abreviada**]).

A continuación se muestran algunos ejemplos de rutas de ubicación que utilizan la sintaxis no abreviada:

- `child::para`selecciona los `para`elementos hijos del nodo de contexto
- `child::*`selecciona todos los elementos secundarios del nodo de contexto
- `child::text()`selecciona todos los nodos de texto secundarios del nodo de contexto
- `child::node()`selecciona todos los hijos del nodo de contexto, cualquiera que sea su tipo de nodo
- `attribute::name`selecciona el `name`atributo del nodo de contexto
- `attribute::*`selecciona todos los atributos del nodo de contexto
- `descendant::para`selecciona los `para`elementos descendientes del nodo de contexto
- `ancestor::div`selecciona todos `div` los ancestros del nodo de contexto
- `ancestor-or-self::div`selecciona los `div`ancestros del nodo de contexto y, si el nodo de contexto es un `div`elemento, el nodo de contexto también
- `descendant-or-self::para`selecciona los `para`elementos descendientes del nodo de contexto y, si el nodo de contexto es un `para`elemento, el nodo de contexto también
- `self::para`selecciona el nodo de contexto si es un `para`elemento y, en caso contrario, no selecciona nada
- `child::chapter/descendant::para` selecciona los `para`elementos descendientes de los `chapter`elementos hijos del nodo de contexto
- `child::*/child::para`selecciona todos `para`los nietos del nodo de contexto
- `/`selecciona la raíz del documento (que siempre es el padre del elemento del documento)
- `/descendant::para`selecciona todos los `para`elementos en el mismo documento como el nodo de contexto
- `/descendant::olist/child::item`selecciona todos los `item`elementos que tienen un `olist`padre y que están en el mismo documento que el nodo de contexto
- `child::para[position()=1]`selecciona el primer `para`hijo del nodo de contexto

- `child::para[position()=last()]`selecciona el último `para`hijo del nodo de contexto
- `child::para[position()=last()-1]`selecciona el penúltimo `para`hijo del nodo de contexto
- `child::para[position()>1]`selecciona todos los `para`hijos del nodo de contexto distintos del primer `para`hijo del nodo de contexto
- `following-sibling::chapter[position()=1]` selecciona el siguiente `chapter`hermano del nodo de contexto
- `preceding-sibling::chapter[position()=1]` selecciona el `chapter`hermano anterior del nodo de contexto
- `/descendant::figure[position()=42]`selecciona el cuadragésimo segundo `figure`elemento del documento
- `/child::doc/child::chapter[position()=5]/child::section[position()=2]` selecciona el segundo `section`del quinto elemento `chapter`del `doc`documento
- `child::para[attribute::type="warning"]` selecciona todos `para`los hijos del nodo de contexto que tienen un `type`atributo con valor`warning`
- `child::para[attribute::type='warning'][position()=5]` selecciona el quinto `para`hijo del nodo de contexto que tiene un `type`atributo con valor `warning`
- `child::para[position()=5][attribute::type="warning"]` selecciona el quinto `para`hijo del nodo de contexto si ese hijo tiene un `type`atributo con valor `warning`
- `child::chapter[child::title='Introduction']` selecciona los `chapter`hijos del nodo de contexto que tienen uno o más `title`hijos con valor de cadena igual a `Introduction`
- `child::chapter[child::title]`selecciona los `chapter`hijos del nodo de contexto que tienen uno o más `title`hijos
- `child::*[self::chapter or self::appendix]` selecciona los hijos `chapter`y `appendix`del nodo de contexto
- `child::*[self::chapter or self::appendix][position()=last()]`selecciona el último `chapter`o `appendix`hijo del nodo de contexto

Hay dos tipos de rutas de ubicación: rutas de ubicación relativas y rutas de ubicación absoluta.

Una ruta de ubicación relativa consta de una secuencia de uno o más pasos de ubicación separados por `/`. Los pasos en una ruta de ubicación relativa se componen de izquierda a derecha. Cada paso, a su vez, selecciona un conjunto de nodos relativos a un nodo de contexto. Una secuencia inicial de pasos se compone junto con un paso siguiente de la siguiente manera. La secuencia inicial de pasos selecciona un conjunto de nodos relativos a un nodo de contexto. Cada nodo de ese conjunto se utiliza como nodo de contexto para el siguiente paso. Los conjuntos de nodos identificados en ese paso se unen entre sí. El conjunto de nodos identificados por la composición de los pasos es esta unión. Por ejemplo, `child::div/child::para`selecciona los `para`elementos hijos de los `div`elementos hijos del nodo de contexto o, en otras palabras, los `para`elementos nietos que tienen `div` padres.

Una ruta de ubicación absoluta consta de, /opcionalmente, una ruta de ubicación relativa. A /por sí solo selecciona el nodo raíz del documento que contiene el nodo de contexto. Si va seguido de una ruta de ubicación relativa, entonces la ruta de ubicación selecciona el conjunto de nodos que serían seleccionados por la ruta de ubicación relativa con respecto al nodo raíz del documento que contiene el nodo de contexto.

Rutas de ubicación

[1]	UbicaciónRuta	::=	Ruta de ubicación relativa | Ubicación AbsolutaRuta
[2]	Ubicación AbsolutaRuta	::=	'/' ¿ Ruta de ubicación relativa ? | Ruta de ubicación absoluta abreviada
[3]	Ruta de ubicación relativa	::=	Paso | Paso UbicaciónRelativa '/' | Ruta de ubicación relativa abreviada

2.1 Pasos de ubicación

Un paso de ubicación tiene tres partes:

- un eje, que especifica la relación de árbol entre los nodos seleccionados por el paso de ubicación y el nodo de contexto,
- una prueba de nodo, que especifica el tipo de nodo y el nombre expandido de los nodos seleccionados en el paso de ubicación, y
- cero o más predicados, que utilizan expresiones arbitrarias para refinar aún más el conjunto de nodos seleccionados por el paso de ubicación.

La sintaxis para un paso de ubicación es el nombre del eje y la prueba del nodo separados por dos puntos dobles, seguidos de cero o más expresiones, cada una entre corchetes. Por ejemplo, en `child::para[position()=1]`, `child`es el nombre del eje, `para`es la prueba del nodo y `[position()=1]`es un predicado.

El conjunto de nodos seleccionado por el paso de ubicación es el conjunto de nodos que resulta de generar un conjunto de nodos inicial a partir del eje y la prueba de nodos, y luego filtrar ese conjunto de nodos por cada uno de los predicados a su vez.

El conjunto de nodos inicial consta de nodos que tienen la relación con el nodo de contexto especificado por el eje y el tipo de nodo y el nombre expandido especificados por la prueba de nodo. Por ejemplo, un paso de ubicación `descendant::para`selecciona los `para`elementos descendientes del nodo de contexto: `descendant`especifica que cada nodo en el conjunto de nodos inicial debe ser un descendiente del contexto; `para`especifica que cada nodo en el conjunto de nodos inicial debe ser un elemento llamado `para`. Los ejes disponibles se describen en [**2.2 Ejes**] . Las pruebas de nodos disponibles se describen en [**2.3 Pruebas de nodos**] . El significado de algunas pruebas de nodos depende del eje.

El conjunto de nodos inicial se filtra por el primer predicado para generar un nuevo conjunto de nodos; este nuevo conjunto de nodos se filtra utilizando el segundo predicado, y así sucesivamente. El conjunto de nodos final es el conjunto de nodos seleccionado en el paso de ubicación. El eje afecta cómo se evalúa la expresión en cada

predicado y, por lo tanto, la semántica de un predicado se define con respecto a un eje. Ver [**2.4 Predicados**] .

Pasos de ubicación

[4]	Paso	::=	Predicado de prueba de nodo AxisSpecifier * | Paso abreviado
[5]	Especificador de eje	::=	Nombre del eje '::' | AbreviadoAxisSpecifier

2.2 Ejes

Están disponibles los siguientes ejes:

- el `child`eje contiene los hijos del nodo de contexto
- el `descendant`eje contiene los descendientes del nodo de contexto; un descendiente es hijo o hijo de un niño, etc.; por lo tanto, el eje descendiente nunca contiene nodos de atributos o espacios de nombres
- el `parent`eje contiene el padre del nodo de contexto, si hay uno
- el `ancestor`eje contiene los ancestros del nodo de contexto; los ancestros del nodo de contexto consisten en el padre del nodo de contexto y el padre del padre, etc.; por lo tanto, el eje ancestro siempre incluirá el nodo raíz, a menos que el nodo de contexto sea el nodo raíz.
- el `following-sibling`eje contiene todos los siguientes hermanos del nodo de contexto; si el nodo de contexto es un nodo de atributo o un nodo de espacio de nombres, el `following-sibling`eje está vacío
- el `preceding-sibling`eje contiene todos los hermanos anteriores del nodo de contexto; si el nodo de contexto es un nodo de atributo o un nodo de espacio de nombres, el `preceding-sibling` eje está vacío
- el `following`eje contiene todos los nodos en el mismo documento que el nodo de contexto que están después del nodo de contexto en el orden del documento, excluyendo cualquier descendiente y excluyendo los nodos de atributos y los nodos de espacio de nombres.
- el `preceding`eje contiene todos los nodos en el mismo documento que el nodo de contexto que están antes del nodo de contexto en el orden del documento, excluyendo los ancestros y excluyendo los nodos de atributos y los nodos de espacio de nombres.
- el `attribute`eje contiene los atributos del nodo de contexto; el eje estará vacío a menos que el nodo de contexto sea un elemento
- el `namespace`eje contiene los nodos de espacio de nombres del nodo de contexto; el eje estará vacío a menos que el nodo de contexto sea un elemento
- el `self`eje contiene solo el nodo de contexto en sí

- el `descendant-or-self`eje contiene el nodo de contexto y los descendientes del nodo de contexto
- el `ancestor-or-self`eje contiene el nodo de contexto y los antepasados del nodo de contexto; por lo tanto, el eje ancestro siempre incluirá el nodo raíz

  NOTA: Los ejes `ancestor`, `descendant`, y dividen un documento (ignorando los nodos de atributos y espacios de nombres): no se superponen y juntos contienen todos los nodos del documento `following`.`precedingself`

ejes

[6]	Nombre del eje	::=	'antepasado'
			| 'ancestro-o-yo'
			| 'atributo'
			| 'niño'
			| 'descendiente'
			| 'descendiente-o-yo'
			| 'siguiente'
			| 'hermano-siguiente'
			| 'espacio de nombres'
			| 'padre'
			| 'anterior'
			| 'hermano anterior'
			| 'ser'

2.3 Pruebas de nodo

Cada eje tiene un **tipo de nodo principal** . Si un eje puede contener elementos, entonces el tipo de nodo principal es elemento; en caso contrario, es el tipo de nodos que puede contener el eje. De este modo,

- Para el eje de atributos, el tipo de nodo principal es atributo.
- Para el eje del espacio de nombres, el tipo de nodo principal es el espacio de nombres.
- Para otros ejes, el tipo de nodo principal es elemento.

Una prueba de nodo que es un QName es verdadera si y solo si el tipo de nodo (consulte **[5 Modelo de datos]**) es el tipo de nodo principal y tiene un nombre expandido igual al nombre expandido especificado por QName . Por ejemplo, `child::para`selecciona los `para`elementos hijos del nodo de contexto; si el nodo de contexto no tiene `para`hijos, seleccionará un conjunto vacío de nodos. `attribute::href`selecciona el `href`atributo del nodo de contexto; si el nodo de contexto no tiene ningún `href` atributo, seleccionará un conjunto vacío de nodos.

Un QName en la prueba de nodo se expande a un nombre expandido utilizando las declaraciones de espacio de nombres del contexto de expresión. `xmlns`Esta es la misma forma en que se realiza la expansión para los nombres de tipos de elementos en las etiquetas de inicio y fin, excepto que no se usa el espacio de nombres predeterminado declarado con: si QName no tiene un prefijo, entonces el URI del espacio de nombres es nulo (esta es la misma manera los nombres de los atributos se expanden). Es un error si

QName tiene un prefijo para el cual no hay una declaración de espacio de nombres en el contexto de la expresión.

Una prueba de nodo *es verdadera para cualquier nodo del tipo de nodo principal. Por ejemplo, child::*seleccionará todos los elementos secundarios del nodo de contexto y attribute::*seleccionará todos los atributos del nodo de contexto.

Una prueba de nodo puede tener el formato NCName:* . En este caso, el prefijo se expande de la misma manera que con QName , utilizando las declaraciones de espacio de nombres de contexto. Es un error si no hay una declaración de espacio de nombres para el prefijo en el contexto de la expresión. La prueba de nodo será verdadera para cualquier nodo del tipo principal cuyo nombre expandido tenga el URI del espacio de nombres al que se expande el prefijo, independientemente de la parte local del nombre.

La prueba de nodo text()es válida para cualquier nodo de texto. Por ejemplo, child::text()seleccionará los nodos de texto secundarios del nodo de contexto. De manera similar, la prueba de nodo comment()es verdadera para cualquier nodo de comentario y la prueba de nodo processing-instruction()es verdadera para cualquier instrucción de procesamiento. La processing-instruction()prueba puede tener un argumento que sea Literal ; en este caso, es cierto para cualquier instrucción de procesamiento que tenga un nombre igual al valor del Literal .

Una prueba de nodo node()es válida para cualquier nodo de cualquier tipo.

[7]	Prueba de nodo	::=	NombrePrueba
			| Tipo de nodo '(' ')'
			| 'instrucción-de-procesamiento' '(' Literal ')'

2.4 Predicados

Un eje es un eje de avance o un eje de retroceso. Un eje que solo contiene el nodo o nodos de contexto que están después del nodo de contexto en el orden del documento es un eje directo. Un eje que solo contiene el nodo o nodos de contexto que están antes del nodo de contexto en el orden del documento es un eje inverso. Por lo tanto, los ejes ancestro, ancestro o yo, precedente y hermano anterior son ejes inversos; todos los demás ejes son ejes delanteros. Dado que el eje propio siempre contiene como máximo un nodo, no hay diferencia si es un eje directo o inverso. La **posición de proximidad** de un miembro de un conjunto de nodos con respecto a un eje se define como la posición del nodo en el conjunto de nodos ordenado en el orden del documento si el eje es un eje directo y ordenado en el orden inverso del documento si el eje es un eje inverso. La primera posición es 1.

Un predicado filtra un conjunto de nodos con respecto a un eje para producir un nuevo conjunto de nodos. Para cada nodo en el conjunto de nodos que se va a filtrar, PredicateExpr se evalúa con ese nodo como nodo de contexto, con el número de nodos en el conjunto de nodos como tamaño de contexto y con la posición de proximidad del nodo en el nodo. -establecer con respecto al eje como posición de contexto; si PredicateExpr se evalúa como verdadero para ese nodo, el nodo se incluye en el nuevo conjunto de nodos; en caso contrario, no está incluido.

Un PredicateExpr se evalúa evaluando Expr y convirtiendo el resultado a booleano. Si el resultado es un número, el resultado se convertirá en verdadero si el número es igual a la posición del contexto y, en caso contrario, se convertirá en falso; si el resultado no es un número, entonces el resultado se convertirá como mediante una llamada a la función **booleana** . Por tanto, una ruta de ubicación para[3]es equivalente a para[position()=3].

[8]	Predicado	::=	'[' PredicadoExpr ']'
[9]	PredicadoExpr	::=	Expr.

2.5 Sintaxis abreviada

A continuación se muestran algunos ejemplos de rutas de ubicación que utilizan sintaxis abreviada:

- `para`selecciona los `para`elementos hijos del nodo de contexto
- `*`selecciona todos los elementos secundarios del nodo de contexto
- `text()`selecciona todos los nodos de texto secundarios del nodo de contexto
- `@name`selecciona el `name`atributo del nodo de contexto
- `@*`selecciona todos los atributos del nodo de contexto
- `para[1]`selecciona el primer `para`hijo del nodo de contexto
- `para[last()]`selecciona el último `para`hijo del nodo de contexto
- `*/para`selecciona todos `para`los nietos del nodo de contexto
- `/doc/chapter[5]/section[2]`selecciona el segundo `section`del quinto `chapter`del `doc`
- `chapter//para`selecciona los `para`elementos descendientes de los `chapter`elementos hijos del nodo de contexto
- `//para`selecciona todos los `para`descendientes de la raíz del documento y, por lo tanto, selecciona todos los `para`elementos en el mismo documento como el nodo de contexto
- `//olist/item`selecciona todos los `item` elementos en el mismo documento como el nodo de contexto que tiene un `olist`padre
- `.`selecciona el nodo de contexto
- `.//para`selecciona los `para`elementos descendientes del nodo de contexto
- `..`selecciona el padre del nodo de contexto
- `../@lang`selecciona el `lang`atributo del padre del nodo de contexto
- `para[@type="warning"]`selecciona todos `para` los hijos del nodo de contexto que tienen un `type`atributo con valor`warning`
- `para[@type="warning"][5]`selecciona el quinto `para`hijo del nodo de contexto que tiene un `type` atributo con valor`warning`
- `para[5][@type="warning"]`selecciona el quinto `para`hijo del nodo de contexto si ese hijo tiene un `type`atributo con valor`warning`

- `chapter[title="Introduction"]`selecciona los `chapter`hijos del nodo de contexto que tienen uno o más `title`hijos con [valor de cadena] igual a `Introduction`
- `chapter[title]`selecciona los `chapter` hijos del nodo de contexto que tienen uno o más `title` hijos
- `employee[@secretary and @assistant]`selecciona todos los `employee`hijos del nodo de contexto que tienen tanto un `secretary`atributo como un `assistant` atributo

La abreviatura más importante es la que `child::`se puede omitir en un paso de ubicación. En efecto, `child`es el eje predeterminado. Por ejemplo, una ruta de ubicación `div/para`es la abreviatura de `child::div/child::para`.

También hay una abreviatura para atributos: `attribute::`se puede abreviar como `@`. Por ejemplo, una ruta de ubicación `para[@type="warning"]`es la abreviatura de `child::para[attribute::type="warning"]`y, por lo tanto, selecciona `para`elementos secundarios con un `type`atributo con valor igual a `warning`.

`//`es la abreviatura de `/descendant-or-self::node()/`. Por ejemplo, `//para`es la abreviatura de `/descendant-or-self::node()/child::para`y, por lo tanto, seleccionará cualquier `para`elemento del documento (incluso un `para`elemento que sea un elemento del documento será seleccionado `//para`porque el nodo del elemento del documento es hijo del nodo raíz); `div//para`es la abreviatura de `div/descendant-or-self::node()/child::para`y así seleccionará a todos `para`los descendientes de `div` los niños.

> **NOTA:** La ruta de ubicación `//para[1]`no *significa* lo mismo que la ruta de ubicación `/descendant::para[1]`. Este último selecciona el primer `para`elemento descendiente; el primero selecciona todos `para`los elementos descendientes que son los primeros `para` hijos de sus padres.

Un paso de ubicación `.`es la abreviatura de `self::node()`. Esto es particularmente útil junto con `//`. Por ejemplo, la ruta de ubicación `.//para`es la abreviatura de

```
self::nodo()/descendiente-o-self::nodo()/child::para
```

y así seleccionará todos `para`los elementos descendientes del nodo de contexto.

De manera similar, un paso de ubicación `..`es la abreviatura de `parent::node()`. Por ejemplo, `../title`es la abreviatura de `parent::node()/child::title`y por eso seleccionará los `title`hijos del padre del nodo de contexto.

Abreviaturas

[10]	Ruta de ubicación absoluta abreviada	::=	'//' [RutaUbicaciónRelativa]
[11]	Ruta de ubicación relativa abreviada	::=	[Paso RelativeLocationPath] '//'
[12]	Paso abreviado	::=	'.' | '..'
[13]	AbreviadoAxisSpecifier	::=	¿'@'?

3 expresiones

3.1 Conceptos básicos

Una VariableReference evalúa el valor al que está vinculado el nombre de la variable en el conjunto de enlaces de variables en el contexto. Es un error si el nombre de la variable no está vinculado a ningún valor en el conjunto de vinculaciones de variables en el contexto de la expresión.

Se pueden utilizar paréntesis para agrupar.

[14]	Expr.	::=	OrExpr
[15]	Exprprimaria	::=	Referencia variable
			| '(' Expr ')'
			| Literal
			| Number
			| FunctionCall

3.2 Function Calls

A FunctionCall expression is evaluated by using the FunctionName to identify a function in the expression evaluation context function library, evaluating each of the Arguments, converting each argument to the type required by the function, and finally calling the function, passing it the converted arguments. It is an error if the number of arguments is wrong or if an argument cannot be converted to the required type. The result of the FunctionCall expression is the result returned by the function.

An argument is converted to type string as if by calling the **string** function. An argument is converted to type number as if by calling the **number** function. An argument is converted to type boolean as if by calling the **boolean** function. An argument that is not of type node-set cannot be converted to a node-set.

[16]	FunctionCall	::=	FunctionName '(' (Argument (',' Argument)*)? ')'
[17]	Argument	::=	Expr

3.3 Node-sets

A location path can be used as an expression. The expression returns the set of nodes selected by the path.

The `|` operator computes the union of its operands, which must be node-sets.

Predicates are used to filter expressions in the same way that they are used in location paths. It is an error if the expression to be filtered does not evaluate to a node-set. The Predicate filters the node-set with respect to the child axis.

> **NOTE:** The meaning of a Predicate depends crucially on which axis applies. For example, `preceding::foo[1]` returns the first `foo` element in *reverse document order*, because the axis that applies to the `[1]` predicate is the preceding axis; by contrast, `(preceding::foo)[1]` returns the first `foo` element in *document order*, because the axis that applies to the `[1]` predicate is the child axis.

The `/` and `//` operators compose an expression and a relative location path. It is an error if the expression does not evaluate to a node-set. The `/` operator does composition in the same way as when `/` is used in a location path. As in location paths, `//` is short for `/descendant-or-self::node()/`.

There are no types of objects that can be converted to node-sets.

[18]	UnionExpr	::=	PathExpr
			| UnionExpr '|' PathExpr
[19]	PathExpr	::=	LocationPath
			| FilterExpr
			| FilterExpr '/' RutaUbicaciónRelativa
			| FilterExpr '//' RutaUbicaciónRelativa
[20]	Exprfiltro	::=	Exprprimaria
			| Predicado FilterExpr

3.4 Booleanos

Un objeto de tipo booleano puede tener uno de dos valores, verdadero y falso.

Una `or`expresión se evalúa evaluando cada operando y convirtiendo su valor a booleano como si fuera una llamada a la función **booleana** . El resultado es verdadero si alguno de los valores es verdadero y falso en caso contrario. El operando derecho no se evalúa si el operando izquierdo se evalúa como verdadero.

Una `and`expresión se evalúa evaluando cada operando y convirtiendo su valor a booleano como si fuera una llamada a la función **booleana** . El resultado es verdadero si ambos valores son verdaderos y falso en caso contrario. El operando derecho no se evalúa si el operando izquierdo se evalúa como falso.

Una EqualityExpr (que no es solo una RelationalExpr) o una RelationalExpr (que no es solo una AdditiveExpr) se evalúa comparando los objetos que resultan de la evaluación de los dos operandos. La comparación de los objetos resultantes se define en los tres párrafos siguientes. Primero, las comparaciones que involucran conjuntos de nodos se definen en términos de comparaciones que no involucran conjuntos de nodos; esto se define uniformemente para =, !=, <=, <y >=. >En segundo lugar, las comparaciones que no involucran conjuntos de nodos se definen para =y !=. En tercer lugar, se definen comparaciones que no involucran conjuntos de nodos para <=, y .<>=>

Si ambos objetos a comparar son conjuntos de nodos, entonces la comparación será verdadera si y sólo si hay un nodo en el primer conjunto de nodos y un nodo en el segundo conjunto de nodos de modo que el resultado de realizar la comparación en el El valor de cadena s de los dos nodos es verdadero. Si un objeto a comparar es un conjunto de nodos y el otro es un número, entonces la comparación será verdadera si y sólo si hay un nodo en el conjunto de nodos tal que el resultado de realizar la comparación en el número a ser comparado y según el resultado de convertir el valor de cadena de ese nodo en un número usando la función **numérica** es verdadero. Si un objeto a comparar es un conjunto de nodos y el otro es una cadena, entonces la comparación será verdadera si y sólo si hay un nodo en el conjunto de nodos tal que el resultado de realizar la comparación en el valor de cadena del nodo y la otra cadena es verdadera. Si un objeto a comparar es un conjunto de nodos y el otro es un booleano, entonces la comparación será verdadera si y sólo si el resultado de realizar la comparación en el booleano y en el resultado de convertir el conjunto de nodos en un booleano usar la función **booleana** es verdadero.

Cuando ninguno de los objetos a comparar es un conjunto de nodos y el operador es =o !=, los objetos se comparan convirtiéndolos a un tipo común de la siguiente manera y luego comparándolos. Si al menos un objeto a comparar es booleano, entonces cada

objeto a comparar se convierte a booleano como si se aplicara la función **booleana .** De lo contrario, si al menos un objeto a comparar es un número, entonces cada objeto a comparar se convierte en un número como si se aplicara la función **numérica .** De lo contrario, ambos objetos a comparar se convierten en cadenas como si se aplicara la función **de cadena** . La =comparación será verdadera si y sólo si los objetos son iguales; la != comparación será verdadera si y sólo si los objetos no son iguales. Los números se comparan para determinar la igualdad según IEEE 754 [IEEE 754] . Dos valores booleanos son iguales si ambos son verdaderos o ambos son falsos. Dos cadenas son iguales si y sólo si constan de la misma secuencia de caracteres UCS.

> **NOTA:** Si $xestá vinculado a un conjunto de nodos, entonces $x="foo"no significa lo mismo que not($x!="foo"): lo primero es verdadero si y solo si *algún* nodo $xtiene el valor de cadena foo; lo último es cierto si y sólo si *todos* los nodos $xtienen el valor de cadena foo.

Cuando ninguno de los objetos a comparar es un conjunto de nodos y el operador es <=, o , entonces los objetos se comparan convirtiendo ambos objetos en números y comparando los números de acuerdo con IEEE 754. La <comparación será verdadera si y solo si el primero El número es menor que el segundo número. La comparación será verdadera si y sólo si el primer número es menor o igual que el segundo número. La comparación será verdadera si y sólo si el primer número es mayor que el segundo. La comparación será verdadera si y sólo si el primer número es mayor o igual que el segundo número.>=><<=>>=

> **NOTA:** Cuando aparece una expresión XPath en un documento XML, los operadores any <y <=deben citarse de acuerdo con las reglas XML 1.0 usando, por ejemplo, <y <=. En el siguiente ejemplo, el valor del testatributo es una expresión XPath:

```
<xsl:if prueba="@valor < 10">...</xsl:if>
```

[21]	OrExpr	::=	YExpr
			| OrExpr 'o' AndExpr
[22]	YExpr	::=	IgualdadExpr
			| AndExpr 'y' IgualdadExpr
[23]	IgualdadExpr	::=	Expr relacional
			| ExprIgualdad '=' ExprRelacional
			| ExprIgualdad '!=' ExprRelacional
[24]	Expr relacional	::=	ExprAditivo
			| ExprRelacional '<' ExprAditivo
			| ExprRelacional '>' ExprAditivo
			| ExprRelacional '<=' ExprAditivo
			| ExprRelacional '>=' ExprAditivo

> **NOTA:** El efecto de la gramática anterior es que el orden de precedencia es (la prioridad más baja primero):
>
> - or
> - and
> - =,!=
> - <=, <, >=, >

y todos los operadores son asociativos por izquierda. Por ejemplo, `3 > 2 > 1`es equivalente a `(3 > 2) > 1`, que se evalúa como falso.

3.5 Números

Un número representa un número de punto flotante. Un número puede tener cualquier valor IEEE 754 de formato de 64 bits de doble precisión [IEEE 754] . Estos incluyen un valor especial "No es un número" (NaN), infinito positivo y negativo, y cero positivo y negativo. Consulte la Sección 4.2.3 de [JLS] para obtener un resumen de las reglas clave del estándar IEEE 754.

Los operadores numéricos convierten sus operandos en números como si llamaran a la función **numérica** .

El `+`operador realiza la suma.

El `-`operador realiza la resta.

> **NOTA:** Dado que XML permite `-`la entrada de nombres, el `-` operador normalmente debe ir precedido de un espacio en blanco. Por ejemplo, `foo-bar`se evalúa como un conjunto de nodos que contiene los elementos secundarios denominados `foo-bar`; `foo - bar`se evalúa como la diferencia del resultado de convertir el valor de cadena del primer `foo`elemento secundario en un número y el resultado de convertir el valor de cadena del primer `bar`elemento secundario en un número.

El `div`operador realiza la división en coma flotante según IEEE 754.

El `mod`operador devuelve el resto de una división truncada. Por ejemplo,

- `5 mod 2`devoluciones`1`
- `5 mod -2`devoluciones`1`
- `-5 mod 2`devoluciones`-1`
- `-5 mod -2`devoluciones`-1`

> **NOTA:** Este es el mismo `%`operador en Java y ECMAScript.

> **NOTA:** Esto no es lo mismo que la operación de resto IEEE 754, que devuelve el resto de una división por redondeo.

Expresiones numéricas

[25]	ExprAditivo	::=	Exprmultiplicativa | ExprAditiva '+' ExprMultiplicativa | ExprAditiva '-' ExprMultiplicativa
[26]	Exprmultiplicativa	::=	ExprUnaria | ExprMultiplicativaOperadorMultiplicarExprUnaria _ _ | Exprmultiplicativa ' div'Exprunaria | ExprMultiplicativa ' mod'ExprUnaria
[27]	ExprUnaria	::=	UniónExpr

| '-' ExprUnaria

3.6 cuerdas

Las cadenas constan de una secuencia de cero o más caracteres, donde un carácter se define como en la Recomendación XML [XML] . Por lo tanto, un solo carácter en XPath corresponde a un solo carácter abstracto Unicode con un único valor escalar Unicode correspondiente (consulte [Unicode]); esto no es lo mismo que un valor de código Unicode de 16 bits: la representación de caracteres codificados Unicode para un carácter abstracto con un valor escalar Unicode mayor que U+FFFF es un par de valores de código Unicode de 16 bits (un par sustituto). En muchos lenguajes de programación, una cadena está representada por una secuencia de valores de código Unicode de 16 bits; Las implementaciones de XPath en dichos lenguajes deben tener cuidado de garantizar que un par sustituto se trate correctamente como un único carácter XPath.

> **NOTA:** Es posible que en Unicode haya dos cadenas que deban tratarse como idénticas aunque consten de secuencias distintas de caracteres abstractos Unicode. Por ejemplo, algunos caracteres acentuados pueden representarse en forma precompuesta o descompuesta. Por lo tanto, las expresiones XPath pueden devolver resultados inesperados a menos que tanto los caracteres de la expresión XPath como los del documento XML se hayan normalizado a una forma canónica. Ver [Modelo de personaje] .

3.7 Estructura léxica

Al tokenizar, siempre se devuelve el token más largo posible.

Para facilitar la lectura, se pueden usar espacios en blanco en las expresiones aunque la gramática no lo permita explícitamente: ExprWhitespace se puede agregar libremente dentro de los patrones antes o después de cualquier ExprToken .

Se deben aplicar las siguientes reglas especiales de tokenización en el orden especificado para eliminar la ambigüedad de la gramática ExprToken :

- Si hay un token anterior y el token anterior no es uno de `@`, `::`, `(`, `[`o ,un Operador , entonces a `*`debe reconocerse como MultiplyOperator y NCName debe reconocerse como OperadorName .
- Si el carácter que sigue a NCName (posiblemente después de intervenir ExprWhitespace) es `(`, entonces el token debe reconocerse como NodeType o FunctionName .
- Si los dos caracteres que siguen a NCName (posiblemente después de intervenir ExprWhitespace) son `::`, entonces el token debe reconocerse como AxisName .
- De lo contrario, el token no debe reconocerse como MultiplyOperator , OperadorName , NodeType , FunctionName o AxisName .

Expresión Estructura Léxica

[28]	ExprToken	::=	'(' | ')' | '[' | ']' | '.' | '..' | '@' | ',' | '::' | NombrePrueba | Tipo de nodo

<table>
<tr><td></td><td></td><td></td><td>| Operador
| Nombre de la función
| Nombre del eje
| Literal
| Número
| Referencia variable</td></tr>
<tr><td>[29]</td><td>Literal</td><td>::=</td><td>'"' [^"]* '"'
| "'" [^']* "'"</td></tr>
<tr><td>[30]</td><td>Número</td><td>::=</td><td>¿ Dígitos ('.' Dígitos ?)?
| '.' Dígitos</td></tr>
<tr><td>[31]</td><td>Dígitos</td><td>::=</td><td>[0-9]+</td></tr>
<tr><td>[32]</td><td>Operador</td><td>::=</td><td>Nombre del operador
| Operador Multiplicar
| '/' | '//' | '|' | '+' | '-' | '=' | '!=' | '<' | '<=' | '>' | '>='</td></tr>
<tr><td>[33]</td><td>Nombre del operador</td><td>::=</td><td>'y' | 'o' | 'mod' | 'div'</td></tr>
<tr><td>[34]</td><td>Operador Multiplicar</td><td>::=</td><td>'*'</td></tr>
<tr><td>[35]</td><td>Nombre de la función</td><td>::=</td><td>QName - Tipo de nodo</td></tr>
<tr><td>[36]</td><td>Referencia variable</td><td>::=</td><td>'$' NombreQ</td></tr>
<tr><td>[37]</td><td>NombrePrueba</td><td>::=</td><td>'*'
| NCNombre ':' '*'
| QNombre</td></tr>
<tr><td>[38]</td><td>Tipo de nodo</td><td>::=</td><td>'comentario'
| 'texto'
| 'instrucción-de-procesamiento'
| 'nodo'</td></tr>
<tr><td>[39]</td><td>Expespacio en blanco</td><td>::=</td><td>S</td></tr>
</table>

4 Biblioteca de funciones principales

Esta sección describe funciones que las implementaciones XPath siempre deben incluir en la biblioteca de funciones que se utiliza para evaluar expresiones.

Cada función en la biblioteca de funciones se especifica mediante un prototipo de función, que proporciona el tipo de retorno, el nombre de la función y el tipo de argumentos. Si un tipo de argumento va seguido de un signo de interrogación, entonces el argumento es opcional; de lo contrario, el argumento es necesario.

4.1 Funciones del conjunto de nodos

Función: último *número* ()

La **última** función devuelve un número igual al tamaño del contexto del contexto de evaluación de la expresión.

Función: posición *numérica* ()

La función **de posición** devuelve un número igual a la posición del contexto del contexto de evaluación de la expresión.

Función: recuento *de números* (*conjunto de nodos*)

La función **de conteo** devuelve el número de nodos en el argumento conjunto de nodos.

Función: ID *del conjunto de nodos* (*objeto*)

La función **id** selecciona elementos por su ID única (consulte [**5.2.1 ID únicas**]). Cuando el argumento de **id** es de tipo conjunto de nodos, entonces el resultado es la unión del resultado de aplicar **id** al valor de cadena de cada uno de los nodos en el argumento conjunto de nodos. Cuando el argumento de **id** es de cualquier otro tipo, el argumento se convierte en una cadena como si fuera una llamada a la función **de cadena** ; la cadena se divide en una lista de tokens separados por espacios en blanco (el espacio en blanco es cualquier secuencia de caracteres que coincida con la producción S); el resultado es un conjunto de nodos que contiene los elementos en el mismo documento que el nodo de contexto que tiene una ID única igual a cualquiera de los tokens de la lista.

- `id("foo")`selecciona el elemento con ID único `foo`
- `id("foo")/child::para[position()=5]`selecciona el quinto `para`hijo del elemento con ID única `foo`

Función: *cadena* **nombre-local** (*conjunto de nodos* ?)

La función **de nombre local** devuelve la parte local del nombre expandido del nodo en el argumento conjunto de nodos que es el primero en el orden del documento . Si el argumento conjunto de nodos está vacío o el primer nodo no tiene un nombre expandido , se devuelve una cadena vacía. Si se omite el argumento, el valor predeterminado es un conjunto de nodos con el nodo de contexto como su único miembro.

Función: *cadena* **espacio de nombres-uri** (*conjunto de nodos* ?)

La función **namespace-uri** devuelve el URI del espacio de nombres del nombre expandido del nodo en el argumento node-set que es el primero en el orden del documento . Si el argumento conjunto de nodos está vacío, el primer nodo no tiene un nombre expandido o el URI del espacio de nombres del nombre expandido es nulo, se devuelve una cadena vacía. Si se omite el argumento, el valor predeterminado es un conjunto de nodos con el nodo de contexto como su único miembro.

> **NOTA:** La cadena devuelta por la función **namespace-uri** estará vacía excepto los nodos de elementos y los nodos de atributos.

Función: nombre *de cadena* (¿ *conjunto de nodos* ?)

La función **de nombre** devuelve una cadena que contiene un QName que representa el nombre expandido del nodo en el argumento conjunto de nodos que ocupa el primer lugar en el orden del documento . El QName debe representar el nombre expandido con respecto a las declaraciones de espacio de nombres vigentes en el nodo cuyo nombre expandido se representa. Normalmente, este será el QName que apareció en el origen XML. Este no tiene por qué ser el caso si hay declaraciones de espacios de nombres vigentes en el nodo que asocian varios prefijos con el mismo espacio de nombres. Sin embargo, una implementación puede incluir información sobre el prefijo original en su representación de nodos; en este caso, una implementación puede garantizar que la cadena devuelta sea siempre la misma que el QName utilizado en el código fuente XML. Si el argumento conjunto de nodos está vacío o el primer nodo no tiene un nombre expandido , se devuelve una cadena vacía. Si omitió el argumento, el valor

predeterminado es un conjunto de nodos con el nodo de contexto como su único miembro.

> **NOTA:** La cadena devuelta por la función **de nombre** será la misma que la cadena devuelta por la función **de nombre local** , excepto para los nodos de elementos y los nodos de atributos.

4.2 Funciones de cadena

Función: cadena *cadena* (*objeto* ?)

La función **de cadena** convierte un objeto en una cadena de la siguiente manera:

- Un conjunto de nodos se convierte en una cadena devolviendo el valor de cadena del nodo del conjunto de nodos que ocupa el primer lugar en el orden del documento . Si el conjunto de nodos está vacío, se devuelve una cadena vacía.
- Un número se convierte en una cadena de la siguiente manera
  - NaN se convierte en la cadena`NaN`
  - El cero positivo se convierte en la cadena. `0`
  - el cero negativo se convierte en la cadena `0`
  - el infinito positivo se convierte en la cadena `Infinity`
  - el infinito negativo se convierte en la cadena `-Infinity`
  - si el número es un número entero, el número se representa en forma decimal como un Número sin punto decimal y sin ceros a la izquierda, precedido por un signo menos (`-`) si el número es negativo
  - de lo contrario, el número se representa en forma decimal como un Número que incluye un punto decimal con al menos un dígito antes del punto decimal y al menos un dígito después del punto decimal, precedido por un signo menos (`-`) si el número es negativo; no debe haber ceros a la izquierda antes del punto decimal, aparte posiblemente del dígito requerido inmediatamente antes del punto decimal; más allá del dígito requerido después del punto decimal, debe haber tantos dígitos adicionales, pero solo tantos, como sean necesarios para distinguir de forma única el número de todos los demás valores numéricos IEEE 754.
- El valor booleano falso se convierte en la cadena `false`. El valor verdadero booleano se convierte en la cadena `true`.
- Un objeto de un tipo distinto de los cuatro tipos básicos se convierte en una cadena de una manera que depende de ese tipo.

Si se omite el argumento, el valor predeterminado es un conjunto de nodos con el nodo de contexto como su único miembro.

> **NOTA:** La `string`función no está diseñada para convertir números en cadenas para presentarlas a los usuarios. La `format-number`función y `xsl:number` el elemento en [XSLT] proporcionan esta funcionalidad.

Función: *cadena* **concat** (*cadena* , *cadena* , *cadena* *)

La función **concat** devuelve la concatenación de sus argumentos.

Función: *booleano* **comienza con** (*cadena* , *cadena*)

La función **comienza con** devuelve verdadero si la cadena del primer argumento comienza con la cadena del segundo argumento y, en caso contrario, devuelve falso.

Función: *booleano* **contiene** (*cadena* , *cadena*)

La función **contiene** devuelve verdadero si la cadena del primer argumento contiene la cadena del segundo argumento y, en caso contrario, devuelve falso.

Función: *cadena* **subcadena-antes** (*cadena* , *cadena*)

La función **subcadena antes** devuelve la subcadena de la primera cadena de argumento que precede a la primera aparición de la segunda cadena de argumento en la primera cadena de argumento, o la cadena vacía si la primera cadena de argumento no contiene la segunda cadena de argumento. Por ejemplo, `substring-before("1999/04/01","/")`devuelve `1999`.

Función: *cadena* **subcadena-después** (*cadena* , *cadena*)

La función **subcadena después** devuelve la subcadena de la primera cadena de argumento que sigue a la primera aparición de la segunda cadena de argumento en la primera cadena de argumento, o la cadena vacía si la primera cadena de argumento no contiene la segunda cadena de argumento. Por ejemplo, `substring-after("1999/04/01","/")`devuelve `04/01`y `substring-after("1999/04/01","19")`devuelve `99/04/01`.

Función: subcadena *de cadena* (*cadena* , *número* , *número* ?)

La función **de subcadena** devuelve la subcadena del primer argumento comenzando en la posición especificada en el segundo argumento con la longitud especificada en el tercer argumento. Por ejemplo, `substring("12345",2,3)`devuelve `"234"`. Si no se especifica el tercer argumento, devuelve la subcadena comenzando en la posición especificada en el segundo argumento y continuando hasta el final de la cadena. Por ejemplo, `substring("12345",2)`devuelve `"2345"`.

Más precisamente, se considera que cada carácter de la cadena (consulte [3.6 Cadenas]) tiene una posición numérica: la posición del primer carácter es 1, la posición del segundo carácter es 2 y así sucesivamente.

> **NOTA:** Esto difiere de Java y ECMAScript, en los que el `String.substring`método trata la posición del primer carácter como 0.

La subcadena devuelta contiene aquellos caracteres para los cuales la posición del carácter es mayor o igual que el valor redondeado del segundo argumento y, si se especifica el tercer argumento, menor que la suma del valor redondeado del segundo argumento y el valor redondeado. valor del tercer argumento; las comparaciones y sumas utilizadas para lo anterior siguen las reglas estándar IEEE 754; el redondeo se realiza como mediante una llamada a la función **de ronda** . Los siguientes ejemplos ilustran varios casos inusuales:

- `substring("12345", 1.5, 2.6)`devoluciones `"234"`
- `substring("12345", 0, 3)`devoluciones `"12"`

- `substring("12345", 0 div 0, 3)`devoluciones `""`
- `substring("12345", 1, 0 div 0)`devoluciones `""`
- `substring("12345", -42, 1 div 0)`devoluciones `"12345"`
- `substring("12345", -1 div 0, 1 div 0)`devoluciones `""`

Función: *número* **longitud de cadena** (*cadena* ?)

La **longitud de la cadena** devuelve el número de caracteres de la cadena (consulte **[3.6 Cadenas]**). Si se omite el argumento, el valor predeterminado es el nodo de contexto convertido en una cadena, en otras palabras, el valor de cadena del nodo de contexto.

Función: *cadena* **normalizar-espacio** (*cadena* ?)

La función **normalize-space** devuelve la cadena de argumento con espacios en blanco normalizados eliminando los espacios en blanco iniciales y finales y reemplazando secuencias de caracteres de espacios en blanco por un solo espacio. Los caracteres de espacio en blanco son los mismos que los permitidos por la producción S en XML. Si se omite el argumento, el valor predeterminado es el nodo de contexto convertido en una cadena, en otras palabras, el valor de cadena del nodo de contexto.

Función: traducción *de cadenas* (*cadena* , *cadena* , *cadena*)

La función **de traducción** devuelve la cadena del primer argumento con las apariciones de caracteres en la cadena del segundo argumento reemplazadas por el carácter en la posición correspondiente en la cadena del tercer argumento. Por ejemplo, `translate("bar","abc","ABC")`devuelve la cadena `BAr`. Si hay un carácter en la cadena del segundo argumento sin ningún carácter en una posición correspondiente en la cadena del tercer argumento (porque la cadena del segundo argumento es más larga que la cadena del tercer argumento), entonces se eliminan las apariciones de ese carácter en la cadena del primer argumento. Por ejemplo, `translate("--aaa--","abc-","ABC")`devuelve `"AAA"`. Si un carácter aparece más de una vez en la cadena del segundo argumento, la primera aparición determina el carácter de reemplazo. Si la cadena del tercer argumento es más larga que la del segundo argumento, se ignoran los caracteres sobrantes.

> **NOTA:** La función **de traducción** no es una solución suficiente para la conversión de mayúsculas y minúsculas en todos los idiomas. Una versión futura de XPath puede proporcionar funciones adicionales para la conversión de casos.

4.3 Funciones booleanas

Función: *booleana* **booleana** (*objeto*)

La función **booleana** convierte su argumento en booleano de la siguiente manera:

- un número es verdadero si y sólo si no es cero positivo ni negativo ni NaN
- un conjunto de nodos es verdadero si y sólo si no está vacío
- una cadena es verdadera si y sólo si su longitud es distinta de cero
- un objeto de un tipo distinto de los cuatro tipos básicos se convierte a un booleano de una manera que depende de ese tipo

Función: *booleano* **no** (*booleano*)

La función **not** devuelve verdadero si su argumento es falso y falso en caso contrario.

Función: *booleano* **verdadero** ()

La función **verdadera** devuelve verdadero.

Función: *booleano* **falso** ()

La función **falsa** devuelve falso.

Función: idioma *booleano* (*cadena*)

La función **lang** devuelve verdadero o falso dependiendo de si el idioma del nodo de contexto especificado por `xml:lang`los atributos es el mismo o es un sublenguaje del idioma especificado por la cadena de argumento. El idioma del nodo de contexto está determinado por el valor del `xml:lang` atributo en el nodo de contexto o, si el nodo de contexto no tiene `xml:lang`atributo, por el valor del `xml:lang`atributo en el ancestro más cercano del nodo de contexto que tiene un `xml:lang`atributo. Si no existe tal atributo, **lang** devuelve falso. Si existe tal atributo, entonces **lang** devuelve verdadero si el valor del atributo es igual al argumento ignorando el caso, o si hay algún sufijo que comienza con `-`tal que el valor del atributo es igual al argumento ignorando ese sufijo del valor del atributo e ignorando caso. Por ejemplo, `lang("en")`devolvería verdadero si el nodo de contexto es cualquiera de estos cinco elementos:

```
<para xml:lang="es"/>
<div xml:lang="es"><para/></div>
<para xml:lang="ES"/>
<para xml:lang="es-es"/>
```

4.4 Funciones numéricas

Función: *número* **número** (*objeto* ?)

La función **numérica** convierte su argumento en un número de la siguiente manera:

- una cadena que consta de espacios en blanco opcionales seguidos de un signo menos opcional seguido de un número seguido de espacios en blanco se convierte al número IEEE 754 más cercano (según la regla de redondeo al más cercano de IEEE 754) al valor matemático representado por cadena; cualquier otra cadena se convierte a NaN
- booleano verdadero se convierte en 1; booleano falso se convierte a 0
- un conjunto de nodos se convierte primero en una cadena como mediante una llamada a la función **de cadena** y luego se convierte de la misma manera que un argumento de cadena
- un objeto de un tipo distinto de los cuatro tipos básicos se convierte en un número de una manera que depende de ese tipo

Si se omite el argumento, el valor predeterminado es un conjunto de nodos con el nodo de contexto como su único miembro.

> **NOTA:** La función **numérica** no debe usarse para la conversión de datos numéricos que ocurren en un elemento de un documento XML a menos que

el elemento sea de un tipo que represente datos numéricos en un formato neutral del idioma (que normalmente se transformaría en un formato específico del idioma). formato para presentación a un usuario). Además, la función **numérica** no se puede utilizar a menos que el formato neutral del idioma utilizado por el elemento sea coherente con la sintaxis XPath para un Número .

Función: suma *de números* (*conjunto de nodos*)

La función **suma** devuelve la suma, para cada nodo en el argumento conjunto de nodos, del resultado de convertir los valores de cadena del nodo en un número.

Función: piso *numérico* (*número*)

La función **suelo** devuelve el número más grande (más cercano al infinito positivo) que no es mayor que el argumento y que es un número entero.

Función: techo *numérico* (*número*)

La función **techo** devuelve el número más pequeño (más cercano al infinito negativo) que no sea menor que el argumento y que sea un número entero.

Función: ronda *numérica* (*número*)

La función **redonda** devuelve el número más cercano al argumento y que es un número entero. Si hay dos de esos números, se devuelve el que esté más cerca del infinito positivo. Si el argumento es NaN, se devuelve NaN. Si el argumento es infinito positivo, entonces se devuelve infinito positivo. Si el argumento es infinito negativo, entonces se devuelve infinito negativo. Si el argumento es cero positivo, entonces se devuelve cero positivo. Si el argumento es cero negativo, se devuelve cero negativo. Si el argumento es menor que cero, pero mayor o igual a -0,5, se devuelve cero negativo.

> **NOTA:** Para estos dos últimos casos, el resultado de llamar a la función **redonda** no es el mismo que el resultado de sumar 0,5 y luego llamar a la función **suelo .**

5 modelo de datos

XPath opera en un documento XML como un árbol. Esta sección describe cómo XPath modela un documento XML como un árbol. Este modelo es sólo conceptual y no exige ninguna implementación en particular. La relación de este modelo con el conjunto de información XML [Conjunto de información XML] se describe en [**B Mapeo de conjunto de información XML**] .

Los documentos XML operados por XPath deben cumplir con la Recomendación de espacios de nombres XML [Nombres XML] .

El árbol contiene nodos. Hay siete tipos de nodos:

- nodos raíz
- nodos de elementos
- nodos de texto
- nodos de atributos

- nodos de espacio de nombres
- procesamiento de nodos de instrucciones
- nodos de comentarios

Para cada tipo de nodo, existe una manera de determinar un **valor de cadena** para un nodo de ese tipo. Para algunos tipos de nodo, el valor de cadena es parte del nodo; para otros tipos de nodos, el valor de cadena se calcula a partir del valor de cadena de los nodos descendientes.

> **NOTA:** Para los nodos de elemento y los nodos raíz, el valor de cadena de un nodo no es el mismo que la cadena devuelta por el `nodeValue`método DOM (consulte [DOM]).

Algunos tipos de nodos también tienen un **nombre expandido** , que es un par que consta de una parte local y un URI de espacio de nombres. La parte local es una cadena. El URI del espacio de nombres es nulo o una cadena. El URI del espacio de nombres especificado en el documento XML puede ser una referencia de URI como se define en [RFC2396] ; esto significa que puede tener un identificador de fragmento y puede ser relativo. Un URI relativo debe resolverse en un URI absoluto durante el procesamiento del espacio de nombres: los URI de espacio de nombres de los nombres expandidos de los nodos en el modelo de datos deben ser absolutos. Dos nombres expandidos son iguales si tienen la misma parte local y ambos tienen un URI de espacio de nombres nulo o ambos tienen URI de espacio de nombres no nulos que son iguales.

Hay un orden, **orden de los documentos** , definido en todos los nodos del documento correspondiente al orden en que aparece el primer carácter de la representación XML de cada nodo en la representación XML del documento después de la expansión de las entidades generales. Por tanto, el nodo raíz será el primer nodo. Los nodos de elementos ocurren antes que sus hijos. Por lo tanto, el orden del documento ordena los nodos de elementos en el orden de aparición de su etiqueta de inicio en el XML (después de la expansión de las entidades). Los nodos de atributos y los nodos de espacio de nombres de un elemento aparecen antes que los hijos del elemento. Los nodos del espacio de nombres están definidos para aparecer antes de los nodos de atributos. El orden relativo de los nodos del espacio de nombres depende de la implementación. El orden relativo de los nodos de atributos depende de la implementación.**El orden inverso de los documentos** es el orden inverso del documento .

Los nodos raíz y los nodos de elemento tienen una lista ordenada de nodos secundarios. Los nodos nunca comparten hijos: si un nodo no es el mismo nodo que otro nodo, entonces ninguno de los hijos de un nodo será el mismo nodo que cualquiera de los hijos de otro nodo. Cada nodo que no sea el nodo raíz tiene exactamente un **padre** , que es un nodo de elemento o el nodo raíz. Un nodo raíz o un nodo elemento es el padre de cada uno de sus nodos secundarios.Los **descendientes** de un nodo son los hijos del nodo y los descendientes de los hijos del nodo.

5.1 Nodo raíz

El nodo raíz es la raíz del árbol. Un nodo raíz no existe excepto como raíz del árbol. El nodo de elemento del elemento del documento es hijo del nodo raíz. El nodo raíz también tiene nodos secundarios de procesamiento de instrucciones y comentarios para procesar instrucciones y comentarios que ocurren en el prólogo y después del final del elemento del documento.

El valor de cadena del nodo raíz es la concatenación de los valores de cadena de todos los nodos de texto descendientes del nodo raíz en el orden del documento.

El nodo raíz no tiene un nombre expandido .

5.2 Nodos de elementos

Hay un nodo de elemento para cada elemento del documento. Un nodo de elemento tiene un nombre expandido que se calcula expandiendo el QName del elemento especificado en la etiqueta de acuerdo con la Recomendación de espacios de nombres XML [Nombres XML] . El URI del espacio de nombres del nombre expandido del elemento será nulo si QName no tiene prefijo y no existe un espacio de nombres predeterminado aplicable.

> **NOTA:** En la notación del Apéndice A.3 de [Nombres XML] , la parte local del nombre expandido corresponde al `type`atributo del `ExpEType`elemento; el URI del espacio de nombres del nombre expandido corresponde al `ns` atributo del `ExpEType`elemento y es nulo si se omite el `ns`atributo del `ExpEType`elemento.

Los hijos de un nodo de elemento son los nodos de elemento, los nodos de comentarios, los nodos de instrucciones de procesamiento y los nodos de texto para su contenido. Se amplían las referencias de entidades a entidades internas y externas. Las referencias de personajes están resueltas.

El valor de cadena de un nodo de elemento es la concatenación de los valores de cadena de todos los nodos de texto descendientes del nodo de elemento en el orden del documento.

5.2.1 ID únicas

Un nodo de elemento puede tener un identificador único (ID). Este es el valor del atributo que se declara en la DTD como tipo `ID`. No es posible que dos elementos de un documento tengan el mismo ID único. Si un procesador XML informa que dos elementos de un documento tienen el mismo ID único (lo cual sólo es posible si el documento no es válido), entonces el segundo elemento en el orden del documento debe tratarse como si no tuviera un ID único.

> **NOTA:** Si un documento no tiene una DTD, ningún elemento del documento tendrá una ID única.

5.3 Nodos de atributos

Cada nodo de elemento tiene un conjunto asociado de nodos de atributos; el elemento es el padre de cada uno de estos nodos de atributos; sin embargo, un nodo de atributo no es hijo de su elemento padre.

> **NOTA:** Esto es diferente del DOM, que no trata el elemento que lleva un atributo como padre del atributo (ver [DOM]).

Los elementos nunca comparten nodos de atributos: si un nodo de elemento no es el mismo nodo que otro nodo de elemento, entonces ninguno de los nodos de atributos de ese nodo de elemento será el mismo nodo que los nodos de atributos de otro nodo de elemento.

> **NOTA:** El `=`operador prueba si dos nodos tienen el mismo valor, *no* si son el mismo nodo. Por lo tanto, los atributos de dos elementos diferentes pueden compararse como iguales usando `=`, aunque no sean el mismo nodo.

Un atributo predeterminado se trata del mismo modo que un atributo especificado. Si se declaró un atributo para el tipo de elemento en la DTD, pero el valor predeterminado se declaró como `#IMPLIED`y el atributo no se especificó en el elemento, entonces el conjunto de atributos del elemento no contiene un nodo para el atributo.

Algunos atributos, como `xml:lang`y `xml:space`, tienen la semántica que aplican a todos los elementos que son descendientes del elemento que lleva el atributo, a menos que se anulen con una instancia del mismo atributo en otro elemento descendiente. Sin embargo, esto no afecta dónde aparecen los nodos de atributos en el árbol: un elemento tiene nodos de atributos solo para los atributos que se especificaron explícitamente en la etiqueta de inicio o en la etiqueta de elemento vacío de ese elemento o que se declararon explícitamente en la DTD con un valor predeterminado. valor.

Un nodo de atributo tiene un nombre expandido y un valor de cadena . El nombre expandido se calcula expandiendo el QName especificado en la etiqueta del documento XML de acuerdo con la Recomendación de espacios de nombres XML [Nombres XML] . El URI del espacio de nombres del nombre del atributo será nulo si el QName del atributo no tiene un prefijo.

> **NOTA:** En la notación del Apéndice A.3 de [Nombres XML] , la parte local del nombre expandido corresponde al `name`atributo del `ExpAName`elemento; el URI del espacio de nombres del nombre expandido corresponde al `ns` atributo del `ExpAName`elemento y es nulo si se omite el `ns`atributo del `ExpAName`elemento.

Un nodo de atributo tiene un valor de cadena . El valor de cadena es el valor normalizado según lo especificado por la Recomendación XML [XML] . Un atributo cuyo valor normalizado es una cadena de longitud cero no se trata de manera especial: da como resultado un nodo de atributo cuyo valor de cadena es una cadena de longitud cero.

> **NOTA:** Es posible declarar atributos predeterminados en una DTD externa o en una entidad de parámetro externa. La Recomendación XML no requiere que un procesador XML lea una DTD externa o un parámetro externo a menos que esté validando. Es posible que una hoja de estilo u otra función que suponga que el árbol XPath contiene valores de atributos predeterminados declarados en una DTD externa o entidad de parámetros no funcione con algunos procesadores XML que no validan.

No hay nodos de atributos correspondientes a atributos que declaran espacios de nombres (consulte [Nombres XML]).

5.4 Nodos de espacio de nombres

Cada elemento tiene un conjunto asociado de nodos de espacio de nombres, uno para cada prefijo de espacio de nombres distinto que está dentro del alcance del elemento (incluido el `xml`prefijo, que está implícitamente declarado por la Recomendación de espacios de nombres XML [Nombres XML]) y uno para el espacio de nombres predeterminado si hay uno. está dentro del alcance del elemento. El elemento es el padre de cada uno de estos nodos de espacio de nombres; sin embargo, un nodo de espacio de nombres no es hijo de su elemento padre. Los elementos nunca comparten nodos de espacio de nombres: si un nodo de elemento no es el mismo nodo que otro nodo de elemento, entonces ninguno de los nodos de espacio de nombres de un nodo de

elemento será el mismo nodo que los nodos de espacio de nombres de otro nodo de elemento. Esto significa que un elemento tendrá un nodo de espacio de nombres:

- para cada atributo del elemento cuyo nombre comienza con `xmlns:`;
- para cada atributo de un elemento ancestro cuyo nombre comienza `xmlns:`a menos que el propio elemento o un ancestro más cercano redeclare el prefijo;
- para un `xmlns`atributo, si el elemento o algún ancestro tiene un `xmlns`atributo, y el valor del `xmlns`atributo para el elemento más cercano no está vacío

> **NOTA:** Un atributo `xmlns=""`"anula la declaración" del espacio de nombres predeterminado (consulte [Nombres XML]).

Un nodo de espacio de nombres tiene un nombre expandido : la parte local es el prefijo del espacio de nombres (está vacío si el nodo del espacio de nombres es para el espacio de nombres predeterminado); el URI del espacio de nombres siempre es nulo.

El valor de cadena de un nodo de espacio de nombres es el URI del espacio de nombres que se vincula al prefijo del espacio de nombres; si es relativo, debe resolverse como un URI de espacio de nombres en un nombre expandido .

5.5 Nodos de instrucciones de procesamiento

Hay un nodo de instrucción de procesamiento para cada instrucción de procesamiento, excepto cualquier instrucción de procesamiento que ocurra dentro de la declaración de tipo de documento.

Una instrucción de procesamiento tiene un nombre expandido : la parte local es el objetivo de la instrucción de procesamiento; el URI del espacio de nombres es nulo. El valor de cadena de un nodo de instrucción de procesamiento es la parte de la instrucción de procesamiento que sigue al objetivo y cualquier espacio en blanco. No incluye la terminación `?>`.

> **NOTA:** La declaración XML no es una instrucción de procesamiento. Por lo tanto, no existe ningún nodo de instrucción de procesamiento correspondiente a la declaración XML.

5.6 Nodos de comentarios

Hay un nodo de comentario para cada comentario, excepto cualquier comentario que ocurra dentro de la declaración de tipo de documento.

El valor de cadena del comentario es el contenido del comentario sin incluir la apertura `<!--`ni el cierre `-->`.

Un nodo de comentario no tiene un nombre expandido .

5.7 Nodos de texto

Los datos de caracteres se agrupan en nodos de texto. Se agrupan tantos datos de caracteres como sea posible en cada nodo de texto: un nodo de texto nunca tiene un hermano inmediatamente siguiente o anterior que sea un nodo de texto. El valor de cadena de un nodo de texto son los datos del carácter. Un nodo de texto siempre tiene al menos un carácter de datos.

Cada carácter dentro de una sección CDATA se trata como datos de caracteres. Por lo tanto, `<![CDATA[<]]>`en el documento fuente se tratará de la misma manera que `<`. Ambos darán como resultado un solo <carácter en un nodo de texto en el árbol. Por lo tanto, una sección CDATA se trata como si se eliminara `<![CDATA[`y y cada aparición de y fuera reemplazada por y respectivamente.`]]><&<&`

> **NOTA:** Cuando un nodo de texto que contiene un <carácter se escribe como XML, el <carácter se debe escapar, por ejemplo, usando `<`o incluyéndolo en una sección CDATA.

Los caracteres dentro de comentarios, instrucciones de procesamiento y valores de atributos no producen nodos de texto. Los finales de línea en entidades externas están normalizados a #xA como se especifica en la Recomendación XML [XML] .

Un nodo de texto no tiene un nombre expandido .

6 Conformidad

XPath está pensado principalmente como un componente que puede ser utilizado por otras especificaciones. Por lo tanto, XPath se basa en especificaciones que utilizan XPath (como [XPointer] y [XSLT]) para especificar criterios de conformidad de implementaciones de XPath y no define ningún criterio de conformidad para implementaciones independientes de XPath.

Una referencia

A.1 Referencias normativas

IEEE 754
Instituto de Ingenieros Eléctricos y Electrónicos. *Estándar IEEE para aritmética binaria de coma flotante* . Norma ANSI/IEEE 754-1985.

RFC2396
T. Berners-Lee, R. Fielding y L. Masinter. *Identificadores uniformes de recursos (URI): sintaxis genérica* . IETF RFC 2396. Véase http://www.ietf.org/rfc/rfc2396.txt .

XML
Consorcio Mundial de la red. *Lenguaje de marcado extensible (XML) 1.0.* Recomendación del W3C. Ver http://www.w3.org/TR/1998/REC-xml-19980210

Nombres XML
Consorcio Mundial de la red. *Espacios de nombres en XML*. Recomendación del W3C. Ver http://www.w3.org/TR/REC-xml-names

A.2 Otras referencias

Modelo de personaje
Consorcio Mundial de la red. *Modelo de caracteres para la World Wide Web.* Borrador de trabajo del W3C. Ver http://www.w3.org/TR/WD-charmod

DOMINGO
Consorcio Mundial de la red. *Especificación de nivel 1 del modelo de objetos de documento (DOM)*. Recomendación del W3C. Ver http://www.w3.org/TR/REC-DOM-Level-1

JLS
J. Gosling, B. Joy y G. Steele. *La especificación del lenguaje Java* . Consulte http://java.sun.com/docs/books/jls/index.html .

ISO/CEI 10646
ISO (Organización Internacional de Normalización). *ISO/IEC 10646-1:1993, Tecnología de la información - Conjunto universal de caracteres codificados de octetos múltiples (UCS) - Parte 1: Arquitectura y plano multilingüe básico* . Estándar internacional. Consulte http://www.iso.ch/cate/d18741.html .

TEI
CM Sperberg-McQueen, L. Burnard *Directrices para la codificación e intercambio de texto electrónico* . Consulte http://etext.virginia.edu/TEI.html .

Unicódigo
Consorcio Unicode. *El estándar Unicode* . Consulte http://www.unicode.org/unicode/standard/standard.html .

Conjunto de información XML
Consorcio Mundial de la red. *Conjunto de información XML*. Borrador de trabajo del W3C. Ver http://www.w3.org/TR/xml-infoset

Xpuntero
Consorcio Mundial de la red. *Lenguaje de puntero XML (XPointer).* Borrador de trabajo del W3C. Ver http://www.w3.org/TR/WD-xptr

XQL
J. Robie, J. Lapp, D. Schach. *Lenguaje de consulta XML (XQL)* . Ver http://www.w3.org/TandS/QL/QL98/pp/xql.html

XSLT
Consorcio Mundial de la red. *Transformaciones XSL (XSLT).* Recomendación del W3C. Ver http://www.w3.org/TR/xslt

B Mapeo de conjuntos de información XML (no normativo)

Los nodos en el modelo de datos XPath se pueden derivar de los elementos de información proporcionados por el conjunto de información XML [XML Infoset] de la siguiente manera:

> **NOTA:** Una nueva versión del borrador de trabajo del conjunto de información XML, que reemplazará a la versión del 17 de mayo, estaba a punto de completarse en el momento en que se completó la preparación de esta versión de XPath y se esperaba que se publicara al mismo tiempo o en breve. después del lanzamiento de esta versión de XPath. La asignación se proporciona para esta nueva versión del borrador de trabajo del conjunto de información XML. Si la nueva versión del XML Information Set Working aún no ha sido publicada, los miembros del W3C pueden consultar la versión interna del Grupo de Trabajo http://www.w3.org/XML/Group/1999/09/WD-xml-infoset-19990915 .html (solo miembros).

- El nodo raíz proviene del elemento de información del documento. Los hijos del nodo raíz provienen de los *hijos* y de las propiedades *de comentarios de los hijos* .

- Un nodo de elemento proviene de un elemento de información de elemento. Los hijos de un nodo de elemento provienen de las propiedades *hijos* y *comentarios de los hijos* . Los atributos de un nodo de elemento provienen de la propiedad *de atributos* . Los espacios de nombres de un nodo de elemento provienen de la propiedad *de espacios de nombres dentro del alcance* . La parte local del nombre expandido del nodo del elemento proviene de la propiedad *del nombre local* . El URI del espacio de nombres del nombre expandido del nodo del elemento proviene de la propiedad *URI del espacio de nombres* . El ID único del nodo del elemento proviene de la propiedad *secundaria* del elemento de información de atributo en la propiedad *de atributos* que tiene una propiedad *de tipo de atributo* igual a `ID`.

- Un nodo de atributo proviene de un elemento de información de atributo. La parte local del nombre expandido del nodo de atributo proviene de la propiedad *del nombre local* . El URI del espacio de nombres del nombre expandido del nodo de atributo proviene de la propiedad *URI del espacio de nombres* . El valor de cadena del nodo proviene de la concatenación de la propiedad del *código de carácter* de cada miembro de la propiedad *secundaria* .

- Un nodo de texto proviene de una secuencia de uno o más elementos de información de caracteres consecutivos. El valor de cadena del nodo proviene de concatenar la propiedad del *código de carácter* de cada uno de los elementos de información de caracteres.

- Un nodo de instrucción de procesamiento proviene de un elemento de información de instrucción de procesamiento. La parte local del nombre expandido del nodo proviene de la propiedad *de destino* . (La parte del URI del espacio de nombres del nombre expandido del nodo es nula). El valor de cadena del nodo proviene de la propiedad *de contenido* . No hay nodos de instrucción de procesamiento para procesar elementos de instrucción que sean hijos del elemento de información de declaración de tipo de documento.

- Un nodo de comentario proviene de un elemento de información de comentario. El valor de cadena del nodo proviene de la propiedad *de contenido* . No hay nodos de comentarios para elementos de información de comentarios que son hijos del elemento de información de declaración de tipo de documento.

- Un nodo de espacio de nombres proviene de un elemento de información de declaración de espacio de nombres. La parte local del nombre expandido del nodo proviene de la propiedad *del prefijo* . (La parte del URI del espacio de nombres del nombre expandido del nodo es nula). El valor de cadena del nodo proviene de la propiedad *URI del espacio de nombres* .